Single Camera Flexible Projection

John Brosz [*] University of Calgary
Faramarz F. Samavati [†] University of Calgary
M. Sheelagh T. Carpendale [‡] University of Calgary
Mario Costa Sousa [§] University of Calgary

Abstract

We introduce a flexible projection framework that is capable of modeling a wide variety of linear, nonlinear, and hand-tailored artistic projections with a single camera. This framework introduces a unified geometry for all of these types of projections using the concept of a flexible viewing volume. With a parametric representation of the viewing volume, we obtain the ability to create curvy volumes, curvy near and far clipping surfaces, and curvy projectors. Through a description of the framework's geometry, we illustrate its capabilities to recreate existing projections and reveal new projection variations. Further, we apply two techniques for rendering the framework's projections: ray casting, and a limited GPU based scanline algorithm that achieves real-time results.

CR Categories: I.3.3 [Computer Graphics]: Image Generation—Viewing Algorithms

Keywords: projection, parametric modeling, non-photorealistic rendering, nonlinear ray casting.

1 Introduction

Projection is fundamental to computer graphics since all three-dimensional scenes must be projected in some manner onto two-dimensional displays. In addition to the standard parallel and perspective projections, there have been a number of alternate projection models proposed, such as fish-eye, telephoto, and hand-tailored artistic projections. However, in spite of the increased awareness of such alternatives, the standard projections remain commonly used because they fit within a well understood geometric framework [Carlbom and Paciorek 1978] that is readily implementable.

As research into the space of possible projection variations expands, an increasing number of viable individual solutions have been discovered. Also, a number of specific frameworks that describe a family of related projections have been proposed. We continue to expand in this latter direction, proposing a generalized, single camera, flexible projection framework (SCFPF) that incorporates previous individual solutions and existing smaller sub-frameworks. SCFPF provides the following features:

- A coherent general geometric framework that can reproduce all single camera 3D to 2D projections that we have encountered. Additionally, without resorting to additional cameras, this framework can reproduce some projections previously created from several cameras as well as reveal new projection possibilities.

[*]e-mail: brosz@cpsc.ucalgary.ca
[†]e-mail:samavati@cpsc.ucalgary.ca
[‡]e-mail:sheelagh@cpsc.ucalgary.ca
[§]e-mail:mario@cpsc.ucalgary.ca

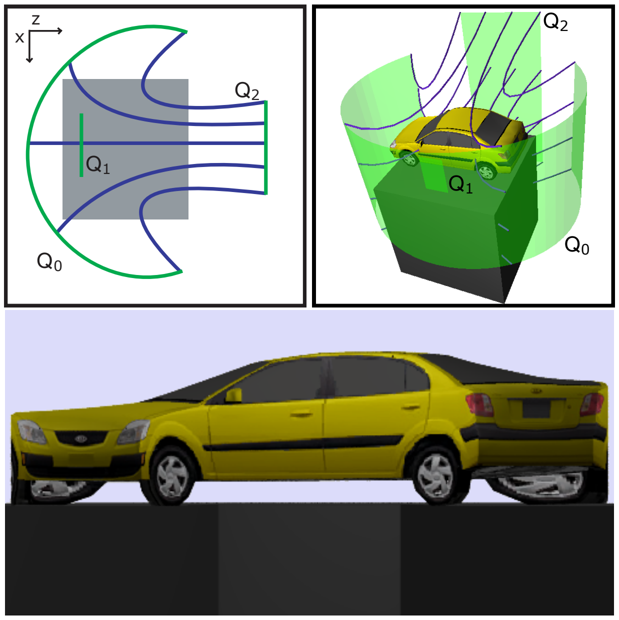

Figure 1: *A projection that uses nonlinear projectors created with our interface. A diagram of the projection and the 3D setup are shown top left and right respectively. The resulting projected image is shown at the bottom. This projection allows us to see almost entirely around the car. Q_0, Q_1, and Q_2 mark the parametric surfaces used to defined the viewing volume.*

- SCFPF geometric models more readily support experimenting with as well as comparing and contrasting different projections.
- The given cohesive geometric descriptions supports subcategorization identifying families within the 3D to 2D projection space that can be supported by our given implementation methods.
- Having a single descriptive camera model provides the basis for offering more comprehensible user access to greater freedoms in projection effects and is a better fit with our daily experience (through use of a single camera as opposed to multiple cameras).

There are many situations where non-standard projections have been useful or effective. Some projections, such as fish-eye and cylindrical panoramic projections, can capture more of the scene than is possible in standard projections. Artistically, projections have frequently been varied from the standard perspective to improve legibility, convey expression [Willats and Durand 2005], create abstraction, assist in shape depiction, and provide impact. To mention just a couple of examples of unique perspective, Cubism images can contain multiple variant perspective views as in Picasso's portrait of Daniel-Henry Kahnweller (1910), or blend varying perspective as in Saint Severin No. 1 by Delaunay (1909). Much more recently, the artistic impact of the animation Ryan [Coleman and Singh 2004], a significant part of which uses blended alternate perspective, has been received with much acclaim.

Although clearly desirable for expanding the capabilities of graphic expression, these non-standard projections are seldom seen in computer generated images. At least in part, this absence may be because non-standard projections tend to be difficult to integrate into the traditional graphics pipeline. The reason for this difficulty is that unlike the standard projections, non-standard projections are often nonlinear in nature and, consequently, are not representable as linear transformations in projection space. When this is the case we categorize it as a *nonlinear projection* as suggested by Salomon [2006]. Another reason for non-standard projections' lack of representation in computer graphics is that previously no common framework existed that was capable of creating all of these different projections. Non-standard projection implementations still tend to be individually hand-tailored. This not only makes it time consuming and expensive to experiment with alternative projections, but makes it nearly impossible for people not familiar with the necessary mathematical background.

Our framework introduces a unified geometry for a wide variety of projections including linear, nonlinear, and artistic projections. This unification makes modeling and rendering of projections easier and assists conceptually in understanding the interplay between the scene, the projection, and the resulting image. Additionally, our system is underlaid by a relatively simple mathematical foundation and does not rely on compositing of individual projections into a single image. In addition to introducing SCFPF, we discuss our implementation choices, including two rendering algorithms: one is based on ray casting; the other uses scanline rendering to achieve realtime performance for a subset of our framework's possible projections.

In the next section, we review a variety of projection techniques that have been proposed in computer graphics. Section 3 defines and describes our framework. Section 4 provides the details of our implementation choices and of rendering SCFPF projections. In Section 5, we show results and discuss how projections from existing works can be reproduced with SCFPF. Lastly, Section 6 presents our conclusions and directions for future work.

2 Related Work

2.1 Single Camera Projections

Wyvill and McNaughton [1990] present Optical Models, a technique for creating projections by mapping an image plane, $\Re^2$, to a set of rays originating from an image surface, $\Re^5$ (three coordinates and two angles). The definition of the mapping is left to the implementation. This technique is capable of reproducing typical camera based projections, such as fish-eye projection, as well as handling curved image planes. Similarly, Glassner [2000; 2004] creates cubist style projections. Like Optical Models, this system is also suited to ray tracing; however, in this system the rays are defined by two NURBS surfaces.

Levene's non-realistic projections [1998] extends Inakage's [1991] non-linear perspective projections and provides the capability to create a variety of artistic projections. This framework uses several parameterized functions that allow users to control the shape of the projection surfaces and the shape of convergence and divergence of orthogonals.

Kolb et al. [1995] developed a technique of reproducing assorted photographic projections by physically simulating lens and camera behaviors. These lens techniques rely on simulating light and consequently use a ray tracing approach.

Most recently, Wang et al. [Wang et al. 2005] presented a volume lens technique using ray casting. The rays are refracted at the image plane based on the specific type of lens selected. A GPU pixel shader then steps through the 3D texture compositing a fragment color based on texels encountered along the ray.

The General Linear Camera (GLC) model described by Yu and McMillan [2004b] is able to reproduce a wide variety of linear projections including perspective, orthogonal, push-broom [Gupta and Hartley 1997], and crossed-slits [Zomet et al. 2003] projections. GLCs are described by three rays. Affine combinations of these defining rays are used to create the rays that sample the scene.

Salomon's book on projections [2006] provides mathematical derivations of a wide variety of linear and nonlinear projections. Wood et al. [1997] and Rademacher and Bishop [1998] have developed systems of creating images that combine more than one viewpoint with a single camera. In Wood et al.'s work, camera paths are used to create a single panoramic image where local areas of the image have the appearance of a perspective projection. Rademacher and Bishop's multiple-center-of-projection images can be described as the result of moving a slit camera along a path through a scene. Mei et al. [2005] introduced the occlusion camera. The projection produced by this camera reveals occluded areas of a 3D objects to assist in image based rendering.

It is important to note that none of these single camera systems is capable of reproducing all of the other single camera systems. In Section 5 we will discuss and demonstrate how these techniques can be created within our framework.

2.2 Composite Projections

There is a rapidly growing body of work that deals with images created by blending the results of two or more cameras' projections together. We refer to these projections as *composite projections*. Works that make use of composite projections include Agrawala et al. [2000], Singh [2002], and Coleman and Singh [2004]. Additionally Popescu et al.'s sampled-based cameras for rendering reflections by blending together many linear camera's projections can be viewed as producing a type of composite projection [2006]. Levene's non-realistic projections [1998], previously mentioned, also includes support for simple composite projections. In general these techniques use multiple cameras positioned throughout the scene. These cameras produce images using linear projections (usually perspective or orthographic) and the main effort of these works lies in blending the images together to produce a coherent composite image as a result. This involves control of the blending process, possibly setting culling distance to keep far-away objects from deforming into the scene, and imposing constraints to maintain geometry.

Singh and Balakrishanan [2004] introduce a projection where local areas of magnification are created by deforming the scene with a FFD deformation lattice. The deformed geometry is then projected to an image.

A variation of composite projection is shown in works by Agarwala et al. [2006], Collomosse and Hall [2003], and Claus and Fitzgibbon [2005]. In these systems, photographs from cameras with unregistered positions and orientations are blended together to create: long images with a variety of vanishing points; non-photorealistic rendering of cubist style paintings; and remove radial distortion, respectively.

Another particular type of composite projection is encompassed by Yu and McMillan's Framework for Multiperspective Rendering [2004a]. In this work the final image is created from a patchwork of tiles; each tile created by a different GLC. By controlling the types and parameters of cameras whose tiles neighbor one another C_0 continuity in the resulting image is guaranteed.

In our system, we have chosen to concentrate on creating projections with a single camera. This avoids the difficult process of blending images together. It also yields an easily visualizable viewing volume that assists in understanding the projection and fits well into our daily experience. Despite using a single camera our framework can reproduce some composite projection effects by bending or splitting the image plane as is shown in Section 5.

3 Projection Framework

The current techniques for specifying projections in computer graphics involve deriving unique matrices from the type of projection (i.e., parallel or perspective) and the camera specification (i.e., viewing angle, position of clipping planes, etc).

We propose a new technique for specifying projections in which the user models the geometry of the viewing volume. One can think of our framework's camera as starting with an orthogonal projection's viewing volume (Figure 2) made from an elastic material. This viewing volume can then be deformed and manipulated to create the desired volume and projection (Figure 3). The end result can be a curvy volume, with curvy near and far clipping surfaces and curvy projectors. Projectors are defined as curves (or lines) starting at the image plane (or image surface) and ending at the far plane (or surface), passing through all points in the scene that could be projected onto the projector's starting position.

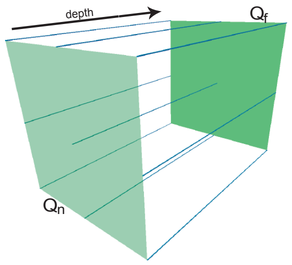

Figure 2: *Viewing volume of an orthogonal projection. We can imagine this as the starting point for SCFPF viewing volumes that can be deformed into various shapes. As will be standard throughout, projectors are shown as blue lines while surfaces within the viewing volume are green. Q_n and Q_f label the near and far planes of the volume.*

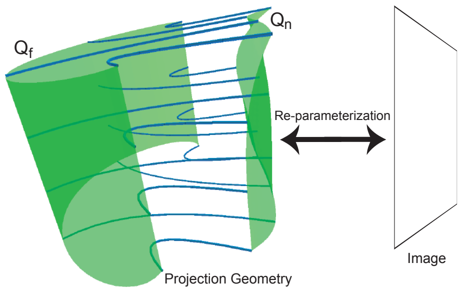

Figure 3: *A complex projection's geometry with nonlinear projectors. Re-parameterization provides a map between Q_n and an image.*

To accomplish this curvy result, we represent a projection's viewing volume as a parametric volume:

$$Q(u,v,t) = \begin{bmatrix} x(u,v,t) \\ y(u,v,t) \\ z(u,v,t) \end{bmatrix}, \quad \begin{matrix} u_0 \le u \le u_1 \\ v_0 \le v \le v_1 \\ 0 \le t \le 1 \end{matrix} .$$

The parameter t corresponds to depth within the projection's viewing volume while u and v identify position within the remaining dimensions (and eventually also the position in the resulting image).

To illustrate $Q(u,v,t)$ let us begin with the orthographic projection shown in Figure 2 and abstract it to a more general setting. For orthogonal projections, the viewing volume is a rectangular prism. From this volume, we can extract two important parametric surfaces, the near plane, $Q_n = Q(u,v,0)$, and the far plane, $Q_f = Q(u,v,1)$. These planes become generalized as surfaces and are parameterized by u and v. Projectors within the orthogonal viewing volume become parallel lines that run between and are orthogonal to the near and far planes. If given an arbitrary point p within this volume, we determine its projection by identifying the u,v coordinates of the projector that intersects p.

Each projector $p_{u,v}(t) = Q(u,v,t)$ within the volume originates at $Q_n(u,v)$ and ends at $Q_f(u,v)$. When creating camera-like projections (e.g., perspective, orthogonal, fish-eye) projectors are linear, composed of rays defined by the two points: $Q_n(u,v)$ and $Q_f(u,v)$. We discuss linear projections further in Section 3.2. To allow full control over all three dimensions of our parameterization, we also allow for nonlinear projectors. That is, projectors that follow a curved path through the 3D scene. The reasons for and effects of allowing this freedom are further discussed in Subsection 3.3. Figure 3 shows an example of a viewing volume with these nonlinear projectors.

One remaining issue is that after we have deformed the viewing volume, our surfaces Q_n and Q_f may no longer be planar. Consequently we may no longer have a concrete viewing plane within the volume. In these cases we use an extra step, that we call re-parameterization, to map projected points from $Q_n(u,v)$ to a viewing plane (see Figure 3). Re-parameterization is described in Subsection 3.4.

To render a given point $p = (x,y,z)$ within the viewing volume we calculate its parametric representation (u,v,t). To accomplish this, we identify the u,v coordinates of the projector that intersects p and then determine its depth along the projector. This step can be expensive in the general case, however there are important cases that lead to simple computations; we will discuss this in Section 4.

In Figure 4, we introduce a diagrammatic representation of the viewing volume. In these diagrams, we present orthogonal views of the viewing volume with attention to the key surfaces, Q_n and Q_f, that can be used to identify the characteristics and behavior of the projection. In addition, represented as blue lines, are a sampling of the viewing volume's projectors. By following the projectors through the volume we can identify where points in the scene will be projected in the image. The surfaces in these diagrams are shown in green and the projectors in blue. Additionally, we will show the camera setup in perspective 3D when we wish to ensure that the original shape of the models is understood and to assist in describing some projections that are not easily described with an orthographic projection (such as the projection shown in Figure 7).

3.1 Relation to Volume Deformation

At this point, our parameterization of the camera's viewing volume might be seen as merely an extension of volume representation or deformation such as free-form deformation (FFD) [Sederberg and Parry 1986] or extended free-form deformation (EFFD) [Coquillart 1990]. To show otherwise we ask you to consider an analogy: the geometry of a pyramid was understood long before it was applied to create the perspective viewing system in traditional art. This development, first scientifically analyzed during the Renaissance constituted a breakthrough for viewing systems. With this in mind, our

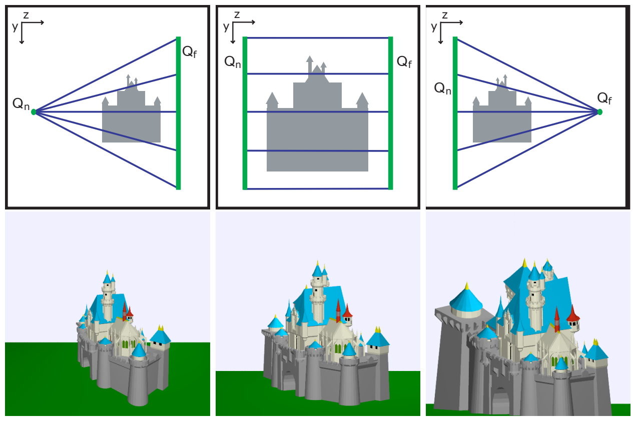

Figure 4: *Creating perspective (left), orthogonal (center), and inverted perspective projections (right). The diagram of the setup of each projection is shown on the left while resulting image is shown on the right.*

goal has been to present the idea of accomplishing projection by deforming the viewing volume. Notice that our goal is not to obtain (or see) deformed objects in the scene; rather it is to observe what happens to the objects after projection (in a nonlinear fashion) to a 2D surface. This creates a variety of new possibilities for creating projections. It is also worthwhile to note that these volume deformation techniques begin with a regular volume, determine the 3D parameterization of objects, and then find the objects' location in the deformed volume. SCFPF is the dual, with the reverse task of determining the objects' parameterization in the deformed volume so the object can be shown within a regular volume (i.e., the viewbox).

3.2 Projection with Linear Projectors

When dealing with linear projectors it is possible to define the entire volume by specifying the two parametric surfaces Q_n and Q_f. It is important to note that these surfaces need not be rectangles or even planes. The volume between these surfaces becomes the viewing volume of the projection. We represent this parametrically as:

$$Q(u,v,t) = (1-t)Q_n(u,v) + tQ_f(u,v)$$
$$t \in [0,1], \quad u \in [u_0,u_1], \quad v \in [v_0,v_1]$$

where u_0, u_1, v_0, and v_1 represent the lower and upper bounds of the parameters used for Q_n and Q_f.

We construct projectors leading from Q_n to Q_f. The definition of the projector that originates at $Q_n(u_0,v_0)$ becomes:

$$q_{u_0,v_0}(t) = (1-t)Q_n(u_0,v_0) + tQ_f(u_0,v_0) \quad t \in [0,1].$$

Points on a particular projector that are at a depth (t) greater than one or less than zero are not included in the image. This provides a far and near clipping distance for each projector.

Now we will show that a perspective projection created by our framework is indeed the perspective projection we are familiar with in computer graphics. In Figure 4 we created the perspective projection using two square patches with normals parallel to the z-axis. Our surface Q_n is a small square and our surface Q_f is a larger square. The center of each square is aligned. These parametric squares are defined as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} c_x + lu \\ c_y + lv \\ c_z \end{bmatrix}$$

where $c = (c_x, c_y, c_z)$ is the center of the square and l is half the width of the square. In our projection we set $c_n = (0,0,1)$ and $c_f = (0,0,2)$, $l_n = 1$, and $l_f = 2$. With Q_n and Q_f defined, our viewing volume is:

$$Q(u,v,t) = (1-t)[c_n + (1,0,0)u + (0,1,0)v] +$$
$$t[c_f + (2,0,0)u + (0,2,0)v].$$

From this we get:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (1-t)u + 2tu \\ (1-t)v + 2tv \\ (1-t) + 2t \end{bmatrix}.$$

We solve for $p^* = (u,v,t)$ and find:

$$\begin{bmatrix} u \\ v \\ t \end{bmatrix} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ z-1 \end{bmatrix}.$$

Let us now look at how we can produce some common projections. By using two square, parallel planes it is simple to reproduce perspective, orthogonal, and inverse perspective projections as is shown in Figure 4. Let us examine our definition of a perspective projection within SCFPF a little more closely. Our surface Q_n is a small parametric rectangle while Q_f is a larger rectangle. In the perspective projection shown in Figure 4 the center of each rectangle is aligned, although this is not always necessary. However, to create a perspective projection, the ratios of width to height of these two rectangles must be the same and the surfaces must be parallel to one another. This ensures that the projectors converge to a single viewing point. By varying the sizes of Q_n and Q_f, as well as their centers, while ensuring that the projectors converge to a single point, we can recreate all possible perspective projections.

It is worth noticing that by shifting Q_n or Q_f and changing their width to height ratios in ways to cause the projectors to no longer converge, we can create entirely new, but somehow related projections. In Figure 5, we present an irregular perspective projection that removes the distortion that is created in the columns perpendicular to the near plane in a wide angle perspective projection. This brief example begins to convey the flexibility and power of our framework's ability to create, modify, and explore the space of possible projections.

One last, pertinent point is in our linear perspective projection, depth of a point in the viewbox, $p = (x,y,z)$, parameterized as (u,v,t) is t. Parameter t can be calculated as $t = \frac{p - Q_n(u,v)}{Q_f(u,v) - Q_n(u,v)}$. This is contrary to most 3D graphics applications [Watt 2000] where linear perspective projection's depth is calculated as: $\bar{t} = \frac{f - d/z}{f - d}$ where f, d, and z are the distances from the center of projection (COP) to the far plane, near plane, and p respectively. This difference can be corrected with a re-parameterization of the depth coordinate t to linear perspective depth $\bar{t}$.

Nonlinear Projections

It is also possible to use linear projectors to create nonlinear projections by creating viewing volumes where Q_n or Q_f are curvy surfaces. For example, by using a small circle (or small hemisphere) as Q_n and a hemisphere centered at that point as Q_f we achieve an angular fish-eye projection [Salomon 2006] (Figure 8). Another popular nonlinear projection is the cylindrical panoramic projection. This is formed by placing one cylinder, Q_n, within another larger cylinder, Q_f (Figure 6).

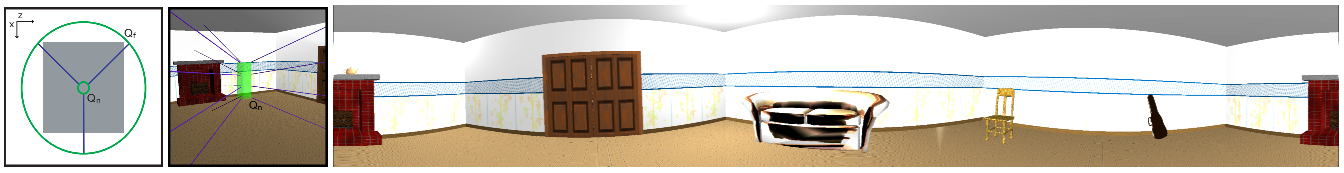

Figure 6: *A cylindrical panorama projection can be created with two nested cylinders.*

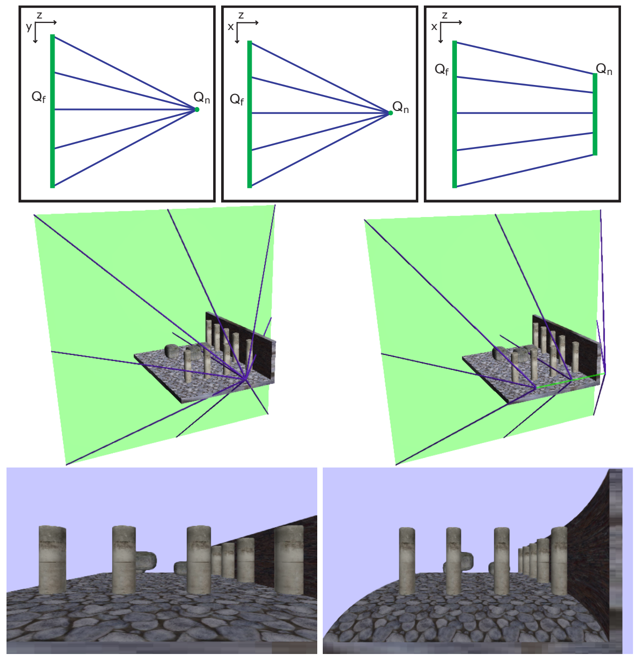

Figure 5: *Comparison between perspective and irregular perspective projections. From left to right the top row of diagrams present the side view of both projections, the top view of perspective, and the top view of irregular perspective. In the middle and bottom rows we present the 3D volume and the results of perspective (left) and irregular perspective (right) projections respectively. Notice that the odd looking distortion of the columns in the perspective result is almost entirely absent in the irregular perspective result.*

The major advantage of this geometric technique for composing projections is that it becomes much easier to visualize, create, and adjust created projections. Rather than dealing with the underlying math, parameters in equations, projections are created through manipulating these relatively simple surfaces and projectors.

3.3 Nonlinear Projectors

By allowing projectors to assume curvy paths between Q_n and Q_f, we can greatly increase the diversity of projections made possible by our framework. Nonlinear projectors allow the nature of the projection to be changed, based on the depth within the viewing volume. The nonlinear projectors are parametric curves, $q_{u_0,v_0}(t)$ for any fixed (u_0, v_0).

In Figure 7, we show an example of the difference in control over the projection made possible by using nonlinear projectors. Nonlinear projectors not only allow more control over the position where a projector intersects an object, but also provide control over the angle at which the projector intersects the object. This is important in creating complicated projections such as those in Figures 11 and 13.

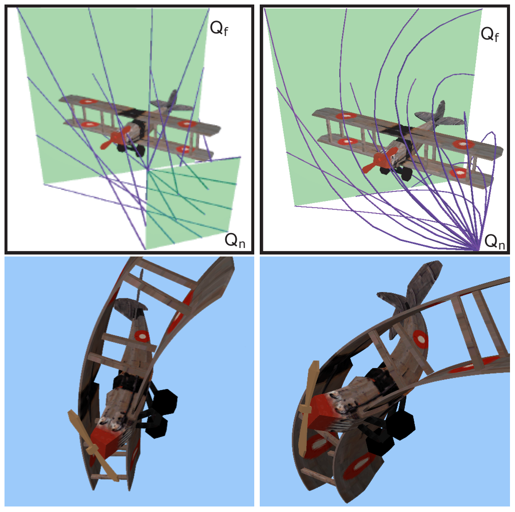

Figure 7: *Comparison of a linear twist projection (left) to a nonlinear projection (right).*

3.4 Re-parameterization

With the viewing volume $Q(u,v,t)$ and a given point in the scene $p = (x,y,z)$ we calculate the projection by finding p's parameterization, $p = (u,v,t)$. Once we have projected all the points, we need to map (u,v,t) to an image. We call this operation re-parameterization. This operation can be visualized as mapping $Q_n(u,v)$ to a viewing surface (screen). Most often this viewing surface will be a plane, but for special applications other surfaces may prove useful. For example, if we had a curved monitor, we could map to the monitor's particular curved shape. When $Q_n(u,v)$ is a parametric surface patch, the re-parameterization is as easy as interpreting (u,v) as the width and height specifications of pixels in the image. However, if we happen to use parametric surfaces based on polar coordinates (e.g., a circle, sphere, or hemisphere) a map from polar coordinates to a rectangular shape in Cartesian coordinates is necessary.

Aside from achieving a flat, rectangular image, re-parameterization can be used to produce other useful effects by resampling the parameterization. A wide variety of possible distortions exist; such distortions can be taken from practically any image space distortion technique such as magic lenses [Yang et al. 2005], or distortion viewing [Carpendale and Montagnese 2001]. One possibility is magnifying a specific local area of the image to produce a magnifying lens effect. Another would be in creating a global distortion that changes an angular fish-eye projection into a lens-like fish-eye. In this distortion, shown in Figure 8, objects nearest the center of the circle appear larger while objects further from the center become compressed.

4 Implementation and Interface

When previously discussing nonlinear projectors, we have left the method for specifying the shape of these curves open. In this sec-

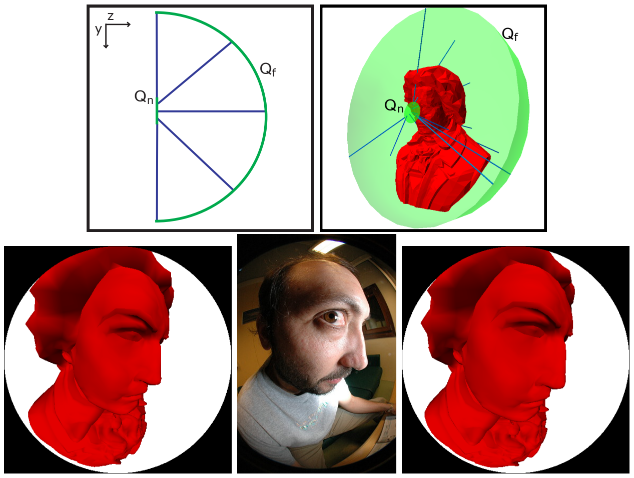

Figure 8: *The top two images present a diagram and the 3D setup of the angular fish-eye projection. In the bottom row, from left to right are the results of an angular fish-eye projection, a picture taken with a fish-eye camera lens ©[Dargaud 2007], and a re-parameterization of the angular fish-eye projection to achieve a lens-like effect.*

tion, we describe one possible technique that helps users to control the flexible viewing volume. This is discussed in subsection 4.1. To render our framework's projections we propose two techniques. The first, presented in subsection 4.2 is for linear projectors and subsection 4.3 for nonlinear projectors is a ray casting approach. Subsection 4.4 presents a scanline rendering algorithm where polygonal coordinates are changed with a vertex shader to positions within the viewbox. This operation achieves realtime results.

4.1 Nonlinear Projector Interface

There are many alternatives for defining our viewing volume, some noteworthy options include implicit definitions and EFFD lattices [Coquillart 1990]. For our implementation, we decided to define our nonlinear projectors as Bezier curves and define the volume with a set of parametric control surfaces. This use of Bezier curves enforces interpolation of Q_n and Q_f so that $Q_n(u,v) = Q(u,v,0)$ and $Q_f(u,v) = Q(u,v,1)$ hold true, and allows us to control the projector curves with control points. Alternatively, B-Spline or NURBS curve schemes could have be used in a similar fashion.

We begin specifying $Q(u,v,t)$ by defining Q_n and Q_f. This provides an initial and final control point for each projector. However, this leaves the remainder of the curve undefined. To provide definition, we first impose the restriction that all curves have the same order and thus the same number of control points. The other control points are provided by creating intermediate control surfaces, an extra surface for each necessary extra control point. To simplify indexing, we now label our surfaces as $Q_0, Q_1, \ldots, Q_m$ where $Q_n = Q_0$ and $Q_f = Q_m$ and $m+1$ is the number of control points required for each projector. These new control surfaces should not be confused with surfaces in the volume; that is $Q_1(u,v)$ is not necessarily equal to $Q(u,v,\frac{1}{m})$. With these control surfaces providing control points, our projector curves are:

$$q_{u,v}(t) = \sum_{i=0}^{m} B_{i,m}(t) Q_i(u,v), \quad t \in [0,1]$$

where $B_{i,m}(t)$ are Bernstein polynomials. Consequently, our viewing volume becomes:

$$Q(u,v,t) = \sum_{i=0}^{m} B_{i,m}(t) Q_i(u,v) \quad u \in [u_0,u_1],\ v \in [v_0,v_1],\ t \in [0,1].$$

With this implementation the user first decides upon the desired type of projectors, determining the appropriate number of control surfaces. These surfaces can come from a list of predefined parametric surfaces (including common options such as a hemisphere, cylinder, bilinear surface, Bezier patch, point, etc) and then have their parameters, positions, and orientations adjusted to create the desired viewing volume. By displaying a few projectors within the volume (as in Figure 1) the viewing volume becomes clearly defined. From this visualization it also becomes relatively easy to predict the behavior of the projection.

If we specify our control surfaces as FFD (or EFFD) control lattices, our use of Bezier curve projectors will cause our entire volume to be parameterized in the same manner as an FFD or EFFD volume. In general, we did not wish to force our control surfaces to be control lattices. The reason for this is that use of non-lattice control surfaces, such as hemispheres and cylinders, allows us to analytically perform ray casting and scanline rendering and avoid resorting to solving with Newton's root finding or similar methods as in [Coquillart 1990].

4.2 Ray Casting

For projections using linear projectors, the implementation is simply that of an ordinary ray caster. The first intersection of rays and objects produces the output image. The projection framework is merely used to create the rays that sample the scene. To trace an individual ray, we first select the image space coordinate (x,y). Next, re-parameterization is used to map these pixel coordinates to our surface coordinates (u,v). The ray that originates at (u,v) is defined by the projector:

$$q_{u,v}(t) = (1-t)Q_n(u,v) + tQ_f(u,v).$$

Intersection tests occur in the same manner as in conventional ray tracers where $t \in [0,1]$ denotes valid intersections occurring within the viewing volume. Lighting, anti-aliasing, acceleration techniques, and other operations, if desired, remain the same as in conventional ray casters.

4.3 Nonlinear Projector Casting

For projections with nonlinear projectors, a more involved approach is necessary. It is clear that the problem of nonlinear projector-object intersection can be much harder than linear projector-object intersection. However, for the class of quadratic and cubic projectors, it is still relatively inexpensive to find these intersections analytically. In this analytical implementation, we have limited our projectors to quadratic curves. Although, this results in projections that are less complex than those with higher order projectors, we have found that quadratics are powerful enough to create a wide variety of projector curves.

The alternative implementation is to use ray segments to approximate the nonlinear projectors. This approach is similar to that of various existing nonlinear ray tracing techniques [Groller 1995; Weiskopf 2000; Weiskopf et al. 2004]. For this type of implementation, $Q(u,v,t)$ can provide a vector field to control the direction of ray segments. Naturally this approach is useful and most accurate when the nonlinear projectors do not deviate greatly from linear projectors. Highly curved projectors however, require a large number of ray segments to approximate in a useful fashion, and consequently may require higher computational costs to achieve accuracy.

A difference between our nonlinear projector casting and ray casting is the question of how to deal with specular highlights when performing Phong lighting calculations. Specular highlights are determined by $(R \cdot V)^n$ where V is the direction to the viewer, R is the light reflection vector, and n denotes the specular power. The problematic term here is V. In our projections, we do not necessarily have a view location or a straight projector (as is the case in Figure 1) to provide this term. Our approach is to use an estimate of the projector's tangent at the point of intersection. This is a reasonable approach as it takes into account the path of the projector at the point in space where it intersects the object.

4.4 Scanline Rendering Algorithm

Our scanline rendering algorithm makes use of the rendering pipeline to produce realtime SCFPF projections. In its current state, this algorithm is only capable of rendering a subset of the projections within our framework.

In this algorithm, we render scenes through our projections with the help of vertex shaders. We use vertex shaders to move vertices from their position in the original geometry to their projected position in the viewbox. Then the existing hardware scanline rendering system renders the triangles of the model. This algorithm is extremely fast although the scanline rendering of triangles can result in inaccuracies in rendering and clipping when coarse models are used. This is due to projection of vertices, rather than all points of each triangle. This problem can be alleviated by subdividing the models in the scene.

The primary task of our scanline algorithm is, for each vertex of the scene, to change its spatial representation (x,y,z) to a parametric representation (u,v,t). Re-parameterization requires an additional step where the u and v values are changed based on the specified re-parameterization mapping.

To solve for (u,v,t), let us start by assuming that for a given $p = (x,y,z)$, $t = t'$ is known where t' is the particular depth value of the vertex within the projection volume (we describe how t' is calculated shortly). The step here is to solve for u and v. Having the description of the volume $Q(u,v,t)$, we are able to find the iso-surface $Q_t'(u,v) = Q(u,v,t')$ that p is located on. Next we for the u and v coordinates for p on $Q_{t'}(u,v)$. The exact details of this solution depends on the definition of $Q_{t'}(u,v)$.

As we are solving for two unknowns (u and v) with three known equations (the parametric equations for x, y, and z that define $Q_{t'}(u,v)$) this algorithm is generally limited to projection geometry where $Q_{t'}(u,v)$ can be defined by first degree equations. Once we have solved for u and v, we then perform the re-parameterization mapping on (u,v).

To complete the calculation of (u,v,t) from (x,y,z), we need to find $t = t'$. Therefore, the parameter t must be calculable with knowledge of $Q(u,v,t)$ and p. Simple examples of where this is possible are the linear projections shown in Figure 4. For these projections, t can be calculated by the ratio of the distance of p to Q_n to the distance between Q_n and Q_f. Other projections for which this algorithm can be used, have all of their u,v iso-surfaces (that is, $Q_t(u,v)$) parallel to one another. We can further extend this to projections with parametric surfaces that are equally distant from one another at all points (i.e., as in the case of nested spheres or cylinders). Examples of possible projections include projections with nonlinear projectors like those in Figures 7 and 9. Figure 1 provides an example of a projection where estimating t is not possible and thus cannot implemented with our scanline technique. Another example of a situation where scanline rendering is not possible is projections where a point (x,y,z) can be mapped to more than one (u,v,t) coordinate.

Our description to this point has been applied to our single camera flexible projection framework in general; however with our specific implementation (using control surfaces) there are some additional considerations. Firstly, if the control surfaces are all the same type of parametric surface, then $Q_{t'}(u,v)$ is easily extracted from $Q(u,v,t)$ as a surface of this same type. The second consideration is that since vertex shaders cannot be overly complex, due to hardware considerations, each type of parametric surface requires a different vertex shader to be implemented to solve for (u,v) given p on the surface.

5 Reproducing Various Projections

Now that we have presented our framework and its implementation we discuss how a variety of projections can be reproduced. We reproduce these projections by discussing how to recreate the geometry of the projections. In some cases additional re-parameterization may be necessary to achieve accurate depth values in the projected coordinates.

When working with linear projectors and without re-parameterization, our framework behaves in the same manner as Optical Models [1990]. Similarly we can reproduce Glassner's Digital Cubism [2000; 2004] by defining our viewing volume through NURBS surfaces. Figures 4, 6, and 8 demonstrate projections that are commonly reproducible by Optical Models, Digital Cubism, and SCFPF. Our technique introduces improvement over these related techniques in three areas. The first is that we introduce nonlinear projectors. These projectors allow the projection to change its characteristics through the depth of the viewing volume. The second is the re-parameterization step that allows the resulting image to be resampled as desired. For example, in Figure 8 it allows us to resample our angular fish-eye to better resemble the photographic fish-eye image. Lastly our technique features a scanline algorithm (albeit a limited one).

As opposed to Kolb et al.'s physical simulation [1995], our goal is not to reproduce all the physical effects present in the camera such as focus, chromatic aberration, etc; but instead to provide a unified method of creating projections. We have shown that we can approximately reproduce a camera's fish-eye projection (Figure 8). Other lens simulations are reproducible with linear projectors that collect the scene's light onto a surface representing the shape of the lens and an appropriate mapping function applied for re-parameterization that mimics the lens' reorganization of light onto the film surface.

In Magic Volumes [Wang et al. 2005] three different projections using linear projectors are described. The first, called the magnification lens, changes the direction of a subset rays to point toward a single point on the far plane, instead of being directed orthogonally. This single point is surrounded by a transition region where the ray direction is pointed slightly toward this single, magnification point. With SCFPF this effect can be produced by using a linear, planar, B-Spline patch with a fine resolution of control points. The magnification point can be created by collapsing several neighboring control points to a single position and by moving surrounding control points closer to this location. The second projection, the sample-rate-based lens, is produced by changing the sampling of the image plane to produce areas of magnification. This is easily accomplished with SCFPF by using an appropriate re-parameterization mapping. Lastly, we have already shown production of the fish-eye lenses that are the third projection presented for Magic Volumes.

In regards to Levene's framework [1998], we have already shown the ability to control convergence and divergence of parallel lines in Figure 4. We can also produce curved convergence of parallel lines as is shown in Figure 9. One particular drawback of Levene's framework is that it is unable to reproduce projections that use a curved projection surface (such as a fish-eye projection).

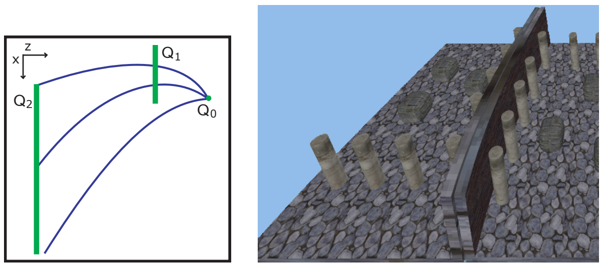

Figure 9: *A projection that uses nonlinear projectors to create curved convergence of parallel lines.*

Next, we would like to show that our framework can reproduce Yu and McMillan's General Linear Cameras (GLC) [2004b]. A GLC is defined by three rays that intersect the image plane at $z = 0$ and another plane at $z = 1$. The rays pass through $z = 0$ at $(u_i, v_i, 0)$ for $i = [1, 2, 3]$ and the $z = 1$ at $(s_i, ti, 1)$. Consequently, each ray can be parameterized in 4D by (s_i, t_i, u_i, v_i). The GLC produces an image by collecting measurements from the affine combinations of the rays:

$$r = \alpha(s_1, t_1, u_1, v_1) + \beta(s_2, t_2, u_2, v_2) + (1 - \alpha - \beta)(s_3, t_3, u_3, v_3).$$

Since α and β simultaneously parameterize both the $z = 0$ and $z = 1$ planes, we can use α and β as the u and v terms per our framework's volume $Q(u, v, t)$. Q_n and Q_f become the planes that bound this volume and, since rays are linear, we use a linear term for t as:

$$Q(u, v, t) = (1 - t)Q_n(u, v) + tQ_f(u, v).$$

Since GLCs can reproduce a variety of linear cameras, this means that SCFPF can produce these cameras as well. Additionally it is clear that GLCs cannot produce nonlinear projectors as any affine combination of linear rays (the rays defining the camera) produces linear rays.

The moving slit camera used in Rademacher and Bishop's [1998] multiple-center-of-projection (MCOP) images can be reproduced using projection with linear projectors. The surface Q_n takes the form of a general cylinder that follows the MCOP path of the camera, while Q_f can be piecewise defined based on the MCOP camera parameters, described as (C_i, O_i, U_i, V_i) for each column of the MCOP image. MCOP projections cannot produce our SCFP as MCOPs' image plane can only curve in one direction due to the use of the slit camera. Additionally MCOP projections do not utilize curved projectors.

The occlusion camera [Mei et al. 2005] can also be reproduced with our flexible projection. This projection uses piecewise linear projectors. These projectors are defined by three control surfaces. The first, Q_0 is a point at the camera position. The second, Q_1 is the plane that Mei et al. refer to as the near distortion plane. The third surface is $Q_3(u, v) = PPHC((u, v, z_f)$ where $PPHC$ is the occlusion camera's projection function for the camera defined by $(u_0, v_0, z_n, z_f, d_n, d_f)$ [Mei et al. 2005]. Figure 10 shows a visualization of an occlusion camera.

In our experiments, we observed that we can also reproduce some kinds of composite projections using our single camera projections.

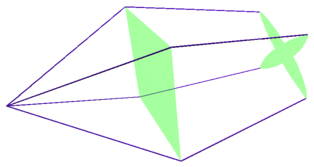

Figure 10: *An occlusion camera [Mei et al. 2005].*

For example, we can start with a regular perspective volume, and adjust the parameterization of Q_1 and Q_2 to direct projectors to approximate separate cameras (Figure 11 presents an example of this). The advantage of this approach is that the nonlinear projectors provide automatic blending between the two cameras.

Figure 11 was create to produce a long street panorama in the spirit of those created by Agarwala et al. [2006]. This image follows a street across the cityscape and, in the result that utilizes nonlinear projectors, is able to present two different perspective views down the regions marked A and B. This example also indicates how a multiperspective panorama by Wood et al. [1997] can be produced. To create even more dramatic multiperspective images Q_n could be changed to follow a curved route through the scene much like MCOP images [Radmacher and Bishop 1998]. Additionally Q_f's height could also be varied to produce areas of differing zoom.

Another composite projection that our framework can produce is Yu and McMillan's Multiperspective Framework (MPF) [Yu and McMillan 2004a]. This is done as follows. For each tile i of the MPF image, we can construct the corresponding GLC with a SCFP volume defined with linear projectors by Q_{n_i} and Q_{f_i}. Our volume for the entire MPF projection is then described by the piecewise control surfaces Q_n and Q_f where each piece of surfaces correspond to the appropriate piece Q_{n_i} and Q_{f_i}.

Let us consider that there is a space of all possible non-composite projections. The related work that we have mentioned covers disjoint subsets and individual solutions within this space. We argue that our framework covers this entire space; not only can we reproduce these individual solutions; but, we can also interpolate between individual solutions and extrapolate to new projections from existing ones. Interpolation is shown in Figure 12 where the created projection is similar to an orthogonal projection at Q_0 but becomes a perspective projection as it approaches Q_2. One can imagine this projection as being useful in a scenario where orthogonal viewing is desired in the area of focus (i.e., near the camera), but displaying the context of the surrounding scene via perspective depth cueing is also wanted. Extrapolation, among other places, is particularly shown in the irregular perspective projection portrayed in Figure 5 where we produce a projection that shares many aspects of perspective projection but is beyond existing projections. Our deformation of the viewing volume is a continuous operation that allows creation of an infinite set of possible projections that includes many undiscovered projections.

In Figure 13, we have attempted to use SCFPF to recreate the projection that M. C. Escher has created in Study for House of Stairs, Pencil and Ink, 1951 [Locker 2000]. Although there are differences between our image and his, this example shows that SCFPF provides the capability of recreating customized artistic projections.

6 Conclusion

We have presented a framework that allows creation of linear, nonlinear, and artistic projections. We have shown that our system is capable of reproducing projections from a variety of existing computer graphics projection techniques.

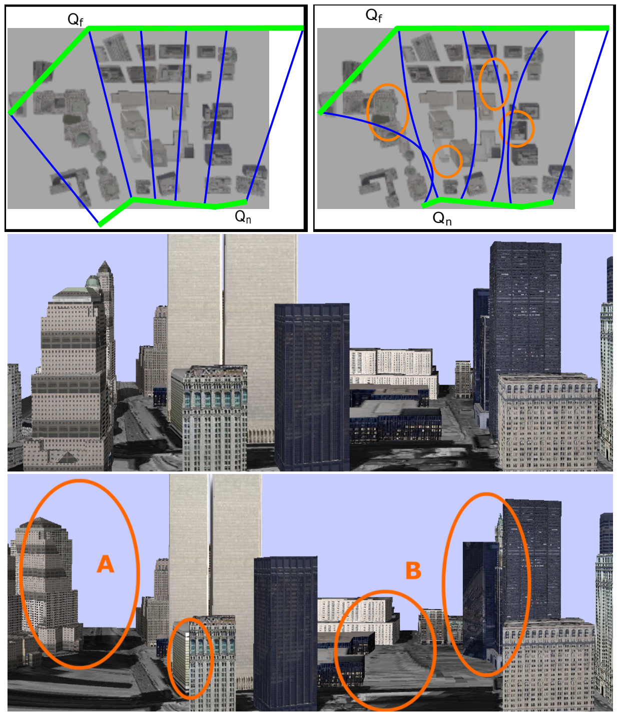

Figure 11: *Effects of nonlinear projectors. At the top we show the construction of two projections, one with linear projectors (left) the other with nonlinear projectors (right). The resulting images produced by these projections are shown in the middle and bottom images. The nonlinear projectors have been used to create perspective viewpoints in areas A and B. Orange ellipses mark the areas of change.*

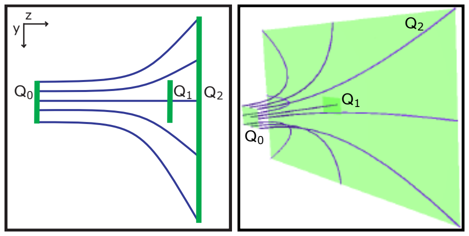

Figure 12: *A projection with nonlinear projectors that is a blend of an orthogonal and a perspective projection.*

We have also introduced nonlinear projectors. While nonlinear projectors have been used in other works, our control of the projectors is the first to not require ray integration (in contrast to nonlinear ray tracing by Groller [1995] and Weiskopf et al. [2004]). Nonlinear projectors are a key element of allowing control over where and at what angle projectors intersect objects.

The most important aspect of this framework is that it allows users to create projections in a geometric and unified fashion. This geometric modeling of projections draws on previous computer modeling experiences and avoids tricky, mathematical, hand-tailored projections. The unified aspect of this system enables us to conceptualize and use a variety of projections in the same way. It is therefore easier to compare and contrast projections with one another.

Lastly, we have described two different rendering implementations: one that makes use of graphics hardware to achieve real time performance, the other based on ray casting.

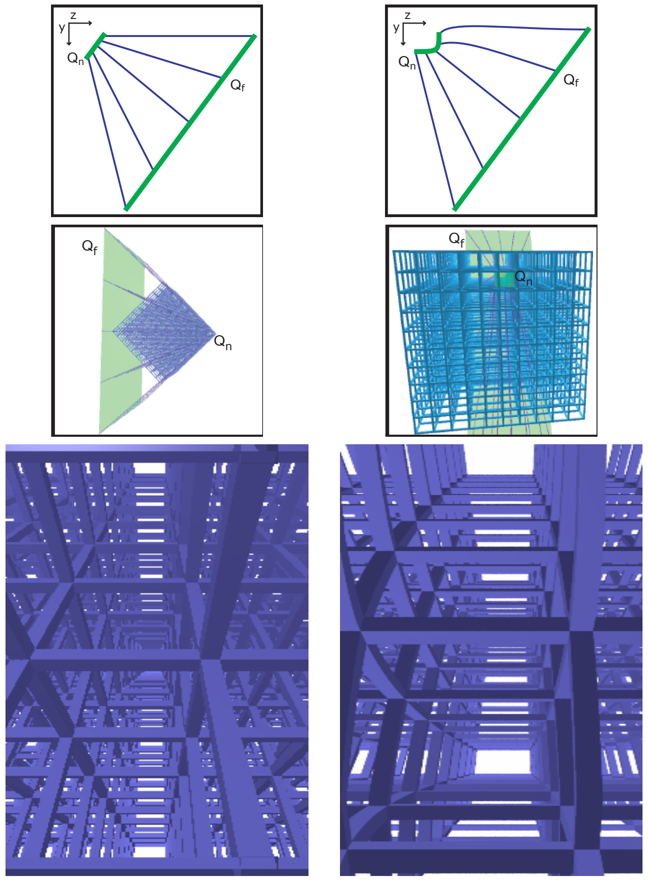

Figure 13: *Recreating the projection used in Study for House of Stairs, Pencil and Ink, 1951 by M. C. Escher [Locker 2000] using SCFPF. On the left we show the details of the setup and result of a perspective projection. On the right we present our designed imitation projection.*

6.1 Limitations and Future Work

A key limitation of our method lies in efficiency. Although we can reproduce a wide variety of projections, including some composite projections, the generality of the framework makes efficient projection difficult. E.g., in MCOP images [Radmacher and Bishop 1998] their composite viewing volume is produced by varying the parameters of strip camera as it follows a path through the scene. However, since a SCFPF control surface may have any shape (and since projectors may be curved) we cannot simply change the parameters of a linear camera across the image. Consequently we rely on ray casting, aside from special cases where scanline rendering is possible.

Our current GPU implementation is limited; creating a less restrictive interactive technique is a major goal in our future work. The casting algorithm also presents interesting challenges. One challenge is adapting ray-based acceleration techniques, such as use of the spatial subdivision, to nonlinear projector intersection tests. Another idea to be addressed is in extending nonlinear projector casting to nonlinear projector tracing; handling of reflections and refraction in a predictable and coherent manner is a key concern. It is our intuition that the distortion of the view volume does not distort the scene's actual geometry and consequently reflections and refractions should be calculated using rays, rather than nonlinear projectors.

Another key area in future work is to examine useful interfaces for creating these projections. Our experience has shown that an interface capable of taking a sketched user input of direction and angle of intersection to create the parameterized viewing volume would greatly assist in creating projections with nonlinear projectors.

Lastly, there should be more work done to directly compare the capabilities of SCFPF and composite projections. Although SCFPF is capable of creating some composite projections (see Figure 11) it is not clear to what extent the two techniques overlap one another. It is likely that the most complete projection system would employ composition of SCFPF projections.

Acknowledgements

We are grateful to Katayoon Etemad for her modeling assistance, to Ruth Hart-Budd and Mark Hancock for their assistance in editing, and the referees for insightful and helpful reviews. The support of the Natural Sciences Research Council of Canada and iCore is appreciatively acknowledged.

References

AGARWALA, A., AGRAWALA, M., COHEN, M., SALESIN, D., AND SZELISKI, R. 2006. Photographing long scenes with multi-viewpoint panoramas. *ACM Transactions on Graphics 25*, 3, 853–861.

AGRAWALA, M., ZORIN, D., AND MUNZNER, T. 2000. Artistic multiprojection rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, Springer-Verlag, 125–136.

CARLBOM, I., AND PACIOREK, J. 1978. Planar geometric projections and viewing transformations. *ACM Comput. Surv. 10*, 4, 465–502.

CARPENDALE, M. S. T., AND MONTAGNESE, C. 2001. A framework for unifying presentation space. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, ACM Press, 61–70.

CLAUS, D., AND FITZGIBBON, A. W. 2005. A rational function lens distortion model for general cameras. In *Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, IEEE Computer Society, vol. 1, 213–219.

COLEMAN, P., AND SINGH, K. 2004. Ryan: rendering your animation nonlinearly projected. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, ACM Press, 129–156.

COLLOMOSSE, J. P., AND HALL, P. M. 2003. Cubist style rendering from photographs. *IEEE Transactions on Visualization and Computer Graphics 9*, 4, 443–453.

COQUILLART, S. 1990. Extended free-form deformation: a culpturing tool for 3d geometric modeling. *Computer Graphics (Proceedings of ACM SIGGRAPH 90) 24*, 4, 187–196.

DARGAUD, G. 2007. Fisheye photography. http://www.gdargaud.net/Photo/Fisheye.html, January.

GLASSNER, A. S. 2000. Cubism and cameras: Free-form optics for computer graphics. Tech. Rep. MSR-TR-2000-05, Microsoft.

GLASSNER, A. S. 2004. Digital cubism. *IEEE Computer Graphics and Applications 24*, 3 (May-Jun), 82–90.

GROLLER, E. 1995. Nonlinear ray tracing: Visualizing strange worlds. *The Visual Computer 11*, 5 (May), 263–274.

GUPTA, R., AND HARTLEY, R. I. 1997. Linear pushbroom cameras. *IEEE Trans. Pattern Anal. Mach. Intell. 19*, 9, 963–975.

INAKAGE, M. 1991. Non-linear perspective projections. In *Modeling in Computer Graphics (Proceedings of the IFIP WG 5.10)*, 203–215.

KOLB, C., MITCHELL, D., AND HANRAHAN, P. 1995. A realistic camera model for computer graphics. In *Proceedings of ACM SIGGRAPH 95*, ACM Press / ACM SIGGRAPH, 317–324.

LEVENE, J. 1998. *A framework for non-realistic projections*. Master's thesis, Massachusetts Institute of Technology.

LOCKER, J. L. 2000. *The Magic of M. C. Escher*. Harry N. Abrams Inc., New York.

MEI, C., POPESCU, V., AND SACKS, E. 2005. The occlusion camera. *Computer Graphics Forum 24*, 3, 335–342.

POPESCU, V., SACKS, E., AND MEI, C. 2006. Sample-based cameras for feed forward reflection rendering. *IEEE Transactions on Visualization and Computer Graphics 12*, 6, 1590–1600.

RADMACHER, P., AND BISHOP, G. 1998. Multiple-center-of-projection images. In *Proceedings of ACM SIGGRAPH 98*, ACM Press / ACM SIGGRAPH, 199–206.

SALOMON, D. 2006. *Transformations and Projections in Computer Graphics*. Springer-Verlag.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, 20, 4, ACM Press, 151–160.

SINGH, K., AND BALAKRISHNAN, R. 2004. Visualizing 3d scenes using non-linear projections and data mining of previous camera movements. In *AFRIGRAPH '04: Proceedings of the 3rd international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, ACM Press, 41–48.

SINGH, K. 2002. A fresh perspective. In *Graphics Interface*, 17–24.

WANG, L., ZHAO, Y., MUELLER, K., AND KAUFMAN, A. 2005. The magic volume lens: An interactive focus+context technique for volume rendering. *VIS 00*, 47.

WATT, A. 2000. *3D Computer Graphics*, third ed. Addison-Wesley Publishing Company Inc.

WEISKOPF, D., SCHAFHITZEL, T., AND ERTL, T. 2004. Gpu-based nonlinear ray tracing. *Computer Graphics Forum 23*, 3, 625–633.

WEISKOPF, D. 2000. Four-dimensional non-linear ray tracing as a visualization tool for gravitational physics. *VIS 00*, 12.

WILLATS, J., AND DURAND, F. 2005. Defining pictorial style: Lessons from linguistics and computer graphics. *Axiomathes 15*, 319–351(33).

WOOD, D. N., FINKELSTEIN, A., HUGHES, J. F., THAYER, C. E., AND SALESIN, D. H. 1997. Multiperspective panoramas for cel animation. In *Proceedings of ACM SIGGRAPH 97*, ACM Press / ACM SIGGRAPH, 243–250.

WYVILL, G., AND MCNAUGHTON, C. 1990. Optical models. In *Proceedings of the eighth international conference of the Computer Graphics Society on CG International '90: computer graphics around the world*, Springer-Verlag, 83–93.

YANG, Y., CHEN, J. X., AND BEHESHTI, M. 2005. Nonlinear perspective projections and magic lenses: 3d view deformation. *IEEE Computer Graphics and Applications 25*, 1, 76–84.

YU, J., AND MCMILLAN, L. 2004. A framework for multiperspective rendering. In *15th Eurographics Symposium on Rendering (EGSR04)*, 61–68.

YU, J., AND MCMILLAN, L. 2004. General linear cameras. In *Computer Vision - ECCV 2004*, Springer Berlin / Heidelberg, vol. 2, 14–27.

ZOMET, A., FELDMAN, D., PELEG, S., AND WEINSHALL, D. 2003. Mosaicing new views: The crossed-slits projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 6, 741–754.